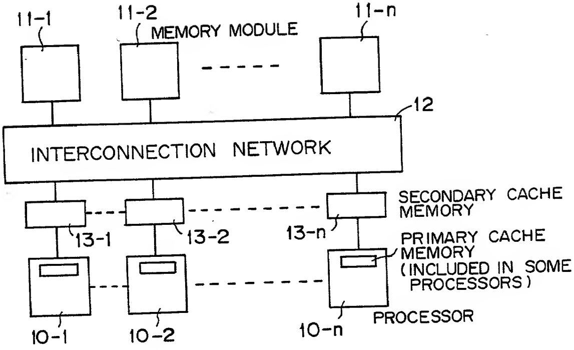

11-1
11-2
MEMORY MODULE
11-n
12
INTERCONNECTION NETWORK
SECONDARY CACHE MEMORY
13-1
13-2
13-n
PRIMARY CACHE MEMORY (INCLUDED IN SOME PROCESSORS)
PROCESSOR
10-1
10-2
10-n

FIG. 1

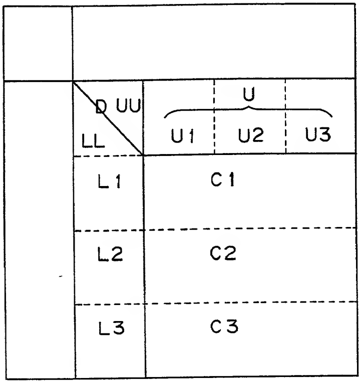


FIG. 2

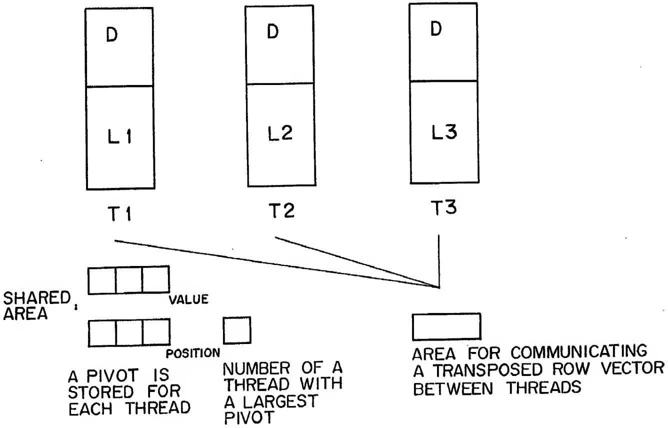

D
L1
T1
D
L2
T2
D
L3
T3
SHARED AREA
VALUE
POSITION
A PIVOT IS STORED FOR EACH THREAD
NUMBER OF A THREAD WITH A LARGEST PIVOT
AREA FOR COMMUNICATING A TRANSPOSED ROW VECTOR BETWEEN THREADS

FIG. 3

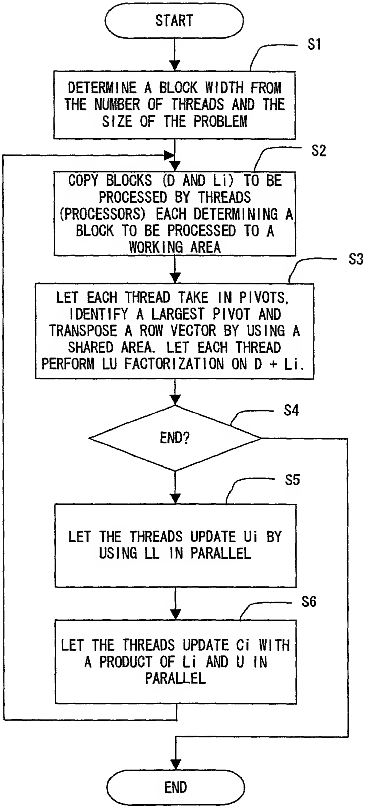

START
S1
DETERMINE A BLOCK WIDTH FROM THE NUMBER OF THREADS AND THE SIZE OF THE PROBLEM
S2
COPY BLOCKS (D AND Li) TO BE PROCESSED BY THREADS (PROCESSORS) EACH DETERMINING A BLOCK TO BE PROCESSED TO A WORKING AREA
S3
LET EACH THREAD TAKE IN PIVOTS, IDENTIFY A LARGEST PIVOT AND TRANSPOSE A ROW VECTOR BY USING A SHARED AREA. LET EACH THREAD PERFORM LU FACTORIZATION ON D + Li.
S4
END?
S5
LET THE THREADS UPDATE Ui BY USING LL IN PARALLEL
S6
LET THE THREADS UPDATE Ci WITH A PRODUCT OF Li AND U IN PARALLEL
END

FIG. 4

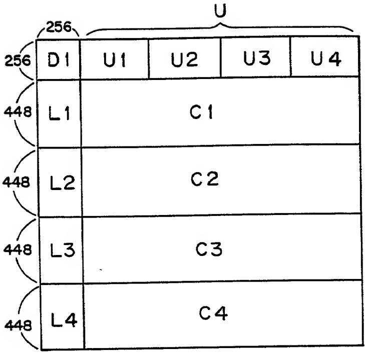


FIG. 5

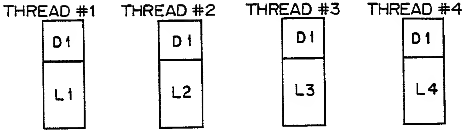
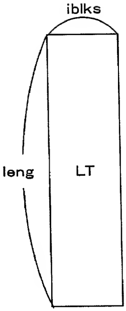
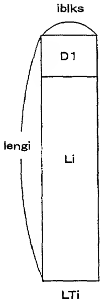


FIG. 6



```
DO  i=1, iblks
 TMP=0、0  DO;jj=0
DO  j=i, leng
 IF(ABS  LT(j, i)), GT   , TMP)THEN
 TMP=ABS(LT(j, i))                              (1)
 jj=j
  ENDIF
ENDDO


IF(jj, GT, i)  THEN
    DO  k=1, iblks
    TMPX=LT(i, k)
    LT(i, k)=LT(jj, k)                         (2)
    LT(jj, k)=TMPX
  ENDDO
END  IF

DO  k=i+1, iblks
    LT(i, k)=LT(i, k)／LT(i, i)
ENDDO

DO  k=i+1, iblks                               (3)
DO  l=i+1, leng
    LT(l, k)=LT(l, k)−LT(l, i)×LT(i, k)
ENDDO
ENDDO

ENDDO
```

FIG. 7



```
DO  i=1, iblks
 TMP=0.0  DO;jj=0
DO  j=1, lengi                                                    ┐
 IF(ABS  LTi(j, i)), GT  , TMP)THEN                               │
 TMP=ABS(LTi(j, i))                                               │ (4)
 jj=i                                                             │
 ENDIF                                                            ┘
ENDDO
pivpot(#THREAD)=jj                                                ┐
(#THREAD IS A THREAD NUMBER.  IN THE                              │
CASE OF PARALLEL PROCESSING BY 4                                  │ (5)
THREADS, #THREAD IS PRESCRIBED AS                                 │
1,2,3 AND 4.)                                                     ┘

BARRIER SYNCHRONIZATION                                           ┐
 iF(#THREAD, EQ, 1)                                               │
    jx=0;GPIVOT=0                                                 │
 DO  ix=1, 4                                                      │
    IF(pivot(ix), GT, jx. AND, PIVOT(ix). GT. iblks)GPIVOT=ix     │ (6)
        (THE NUMBER OF A THREAD HAVING A LARGEST NUMBER)          │
                                                                  │
 ENDDO                                                            │
END  IF                                                           │
BARRIER SYNCHRONIZATION                                           ┘

IF(#THREAD, EQ, GPIVOT)THEN                                       ┐
   IF(jj, GT, i)THEN                                              │
   DO  ix=1, iblks                                                │
   ROW(ix)=LTi(jj, ix)                                            │ (7)
  ENDDO                                                           │
END  IF                                                           │
BARRIER SYNCHRONIZATION                                           ┘
IF(GPIVOT, EQ, O)THEN                                             ┐
   IF(jj, GT, i)THEN                  SINCE TRASPOSITION HAS      │
     DO  i=1, iblks,          <----   BEEN CARRIED OUT IN AN IP,  │
        TMPW=LTi(i, ix)               THE THREADS EXECUTE THE     │
        LTi(i, ix)=LTi(jj, ix)        PROCESSING IN PARALLEL      │
        LTi(jj, ix)=TMPW                                          │
      ENDDO                                                       │
     END  IF                                                      │
ELSE                                                              │
   IF(#THREAD, EQ, GPIVOT)THEN                                    │ (8)
     DO  ix=1, iblks                                              │
        LTi(jj, ix)=LTi(i, ix)                                    │
        LTi(i, ix)=ROW(ix)                                        │
   ENDDO                                                          │
  ELSE                                                            │
     DO  ix=1, iblks                                              │
     LTi(i, ix)=ROW(ix)                                           │
     ENDDO                                                        │
  ENDIF                                                           ┘

   DO  k=i+1, iblks,                                              ┐
          LTi(i, k)=LTi(i, k)/LT(i, i)                            │ (9)
   ENDDO                                                          ┘

   DO  k=i+1, iblks                                               ┐
   DO  l=i+1, lengi                                               │
        LTi(l, k)=LTi(l, k)-LTi(l, i)×LTi(i, k)                   │ (10)
   ENDDO                                                          │
   ENDDO                                                          ┘

   ENDDO
```

F I G. 8

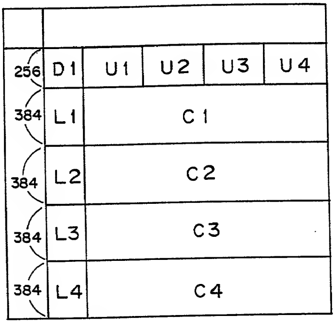


FIG. 9

```
subroutine  LU(LTi, k, iblks, ist, nwid)
   (WHERE LTi IS USED BY THREADS FOR STORING (D1+Li),
     k IS THE SIZE OF THE FIRST ONE DIMENSION OF LTi,
     iblks IS THE BLOCK WIDTH,
     ist IS A POSITION TO START THE Lu FACTORIZATION AND
     nwid IS THE WIDTH OF AN OBJECT SUBJECTED TO THE Lu FACTORIZATION)
 IF(nwid, eq, 8), Then(A WIDTH OF 8 IS A MINIMUM).

LTi(ist:k, ist, ist+nwid-1) IS SUBJECTED TO THE LU FACTORIZATION IN
PARALLEL.
(  HERE, THE PARTS (4) TO (10) OF FIG.9 ARE EXECUTED.
     IN THIS CASE, THE ROW-TRANSPOSING UNIT TRANSPOSES
    LTi(i, 1, iblks) AT THE LENGTH iblk.                      )
else
   call  LU(LTi, k, iblks, ist, nwid/2)
   call  TRS( )
   UPDATE LTi(ist:ist+nwid/2-1, ist+nwid/2:ist+nwid). BY USING A
LOWER-TRIANGULAR MATRIX LL OF LTi(ist:ist+nwid/2-1, ist:ist+nwid/2
-1), UPDATE IT BY MULTIPLYING IT BY LL+ FROM THE LEFT.

   call  MM( )
(  LTi(ist+nwid/2:k, ist+nwid/2:ist+nwid)
   =LTi(ist+nwid/2:k, ist+nwid/2:ist+nwid)
     -LTi(ist+nwid/2:k, ist:ist+nwid/2-1) ×
       LTi(ist:ist+nwid/2-1, ist+nwid/2:ist+nwid)             )

 Barrier SYNCHRONIZATION
 call  LU(LTi, k, iblks, ist+nwid/2, nwid/2
 end  if
    return
    end  subroutine
```

FIG. 10

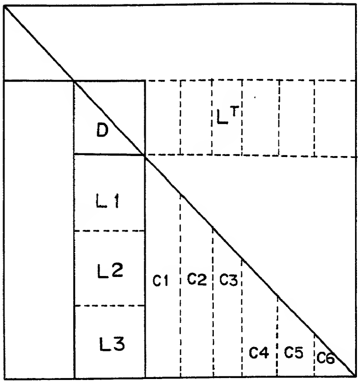


FIG. 11

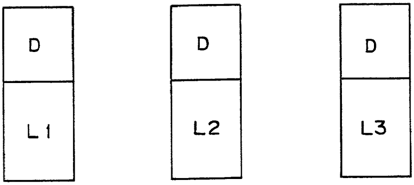


FIG. 12

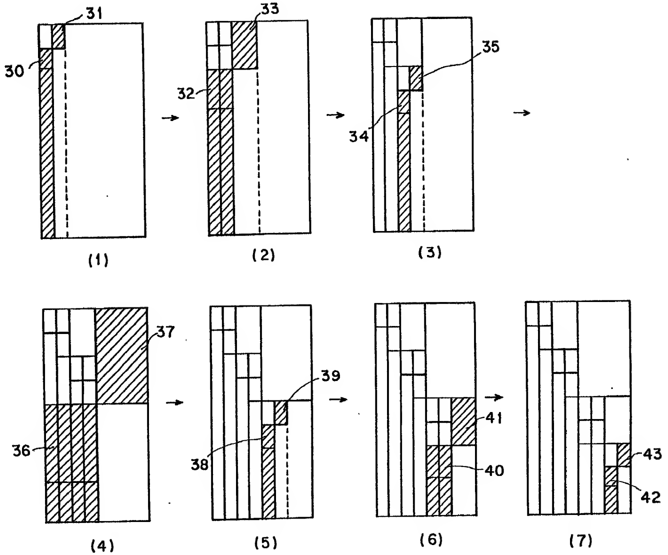


FIG. 13

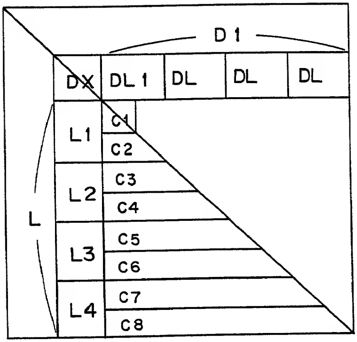


FIG. 14

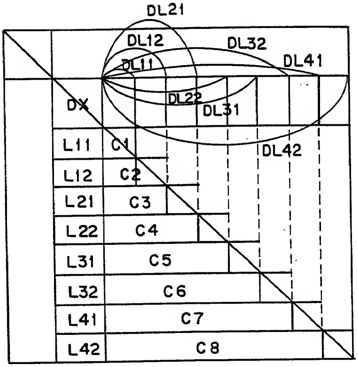

DL21
DL12
DL32
DL11
DL41
DX
DL22
DL31
DL42
L11
C1
L12
C2
L21
C3
L22
C4
L31
C5
L32
C6
L41
C7
L42
C8

FIG. 15

```
subroutine  LTD(LTi, k, iblks, ist, nwid)
 IF(nwid, EQ, 8)THEN    (THE WIDTH OF 8 IS THE MINIMUM)
  DOi=ist, ist+7
   DOj=i+1, ist+7
    LTi(i, j)=LTi(j, i)
    LTi(j, i)=LTi(j, i)/LTi(i, i)
   ENDDO                                                      (20)
   DO  jy=i+1, ist+7
   DO  jx=jx, ist+7
    LTi(jx, jy)=LTi(jx, jy)-LTi(jx, i)×LTi(i, jy)
   ENDDO
   ENDDO

   UPDATE LTi(LTi(ist+8;k, ist:ist+7).
   SINCE $DL^T$ IS INCLUDED IN THE UPPER TRIANGLE OF
    LTi(LTi(ist:ist+7, ist:ist+7), UPDATE $(PL^T)^{-1}$ FROM THE RIGHT.

  ELSE
  call  LDL(LTi, k, iblks, ist, nwid/2)

  COPY $DL^T$ TO
  ·LTi(ist:ist+nwid/2-1, ist+nwid/2:ist+nwid-1).
  (D IS AN OBJECT ELEMENT OF LTi(ist:ist+nwid/2-1, ist:ist+nwid/2-1)
  AND L IS
  LTi(ist+nwid/2:ist+nwid-1, ist:ist+nwid/2-1),
  TRANSPOSING THIS $L^T$.)

   ·UPDATE LTi(ist+nwid/2:k, ist+nwid/2:ist+nwid-1).

   LTi(ist+nwid/2:k, ist+nwid/2:ist+nwid-1)
   =LTi(ist:ist+nwid/2:k, ist+nwid/2:ist+nwid-1)-
   LTi(ist+nwid/2:k, ist:ist+nwid-1)×
   LTi(ist:ist+nwid/2-1, ist+nwid/2;ist+nwid-1)

   CALL  LDL  (LTi, k, iblks, ist+nwid/2, nwid/2)

  ENDIF

   RETURN

   END
```

FIG. 16